Updating the pathway fingerprint

Gabriel Altschuler

July 27, 2011

The fingerprint pipeline involves obtaining a list of the available GEO files, downloading the data, running a pathway enrichment algorithm and then normalizing the results using the GEO corpus as a background. The scripts to run and process the fingerprint are located in the `fingerprint-database-scripts` repository within the `Fingerprint running scripts` or `Fingerprint processing scripts` directories. These scripts were written to run on `hpc111`.

1 Running the fingerprint

1.1 Initiating the run

The script `Updating fingerprint.R` is more of a log file that contains the details of the date and line of code that was used to initiate a fingerprinting run. Scrolling down to the bottom of this script shows the last instance. The `setwd` and `source` lines shown here are the only lines that should be entered into an R terminal, the remaining fingerprint running scripts are designed to run from source and do not need to be 'stepped' through.

1.2 Master script

The script `GEOsurvey_server_update.R` launches and controls the fingerprint running scripts detailed below. The chip dataframe and genesets defined in this script determines the type of arrays that will be surveyed and the pathways sets that will be used to assess enrichment. From here, the

1.3 Obtaining a list of arrays

...

1.4 Donwloading array data

...

1.5 Pathway enrichment analysis

...

2 Post-processing

Once each array has been analyzed using the `single.chip.enrichment` algorithm, the scores must be normalized and combined into a fingerprint data matrix. The script `Fingerprint_post_processing.R` describes the necessary steps, which are detailed here. Running this script is a lengthy process and so is usually run in a `screen`-ed ssh session on hpc111. These scripts are not designed to be sourced as they contain pathnames that need to be updated accordingly and so should be stepped through manually.

2.1 Phase 1

Firstly, the header and file names are specified. This is necessary are there have been many builds of the fingerprint based on different pathway databases and enrichment algorithms. For example, the `fingerpath` directory and `header "/home/galtschu2/Documents/Projects/Fingerprinting/data/Fingerprints/sq/"` and `"sq_"` were used to define the fingerprint dataset run based on the squared-rank score to assess enrichment. Each array is processed and saved as a separate file. The path and header information is used to retrieve the list of relevant filenames. Only files > 200 bytes are used, an indicator that a vector of enrichment scores was successfully constructed. The number of successfully fingerprinted arrays is normally in the range of 150-200,000, with 3-4000 arrays incomplete.

The next step is to remove arrays for this list that are either Agilent based (these 2-color arrays should not be included in the fingerprint) or are based on data that did not contain a full probe set. This occurs when GEO records are incomplete, or in cases when only differentially expressed genes are uploaded. The package `GEOmetadb` is used to retrieve the required metadata.

Once this has been completed, data objects containing the single chip enrichments (SCE) and platform information are constructed by loading each of the validated files. The script contains line to print the array as a rough indication of progress. The resulting dataframes are annotated with the pathway names and saved with a timestamp for reference. Data for each of the platforms is save separately as certain platforms are equivalent and can be combined. The easiest way to combine is to create a new dummy chiptype for the combinations, e.g. the data from GPL96, GPL571, GPL3921 and GPL4685 are combined into the new chiptype HGU133A. This combines normal and high-throughput data for the HGU113A and HGU133A2 chips, as the probe sets are the same. In the next stage, each pathway will be analyzed in turn so normalized the enrichment scores. For efficiency in the subsequent steps of the pipeline, each row of the matrix is saved as a separate pathawy-referenced file.

R does not fully return memory to the system on the server (even after manual garbage collection). This is problematic as the subsequent steps involve parallelization, spawning multiple processes with potentially large memory al-

locations. To avoid this, R is quit and restarted at this stage.

2.2 Phase 2

The distribution of enrichment scores in each pathway is assessed using a method termed *Probability of Expression* (POE). This is described in *Parmigiani et al. A statistical framework for expression based molecular classification in cancer. Journal of the Royal Statistical Society. Series B (Statistical Methodology) (2002) vol. 64 (4).* We will take advantage of the Expectation-Maximization (EM) implementation of POE contained in the R package `metaArray` described in *Choi et al. A latent variable approach for meta-analysis of gene expression data from multiple microarray experiments. BMC bioinformatics (2007) vol. 8 (1).*

Once R has been restarted, a new directory is created to hold the POE data for each platform, for each pathway (i.e. nPlatform x nPathway files). Within each platform the job is parallelized over the pathways across 14 cores using the packages `foreach`, `multicore` and `doMC`. Even with parallelization, calculating the POE across all the platform and pathway combinations is compulationally intensive and this usually runs overnight. Once this has been completed, it is advisable to quit and restart R again to conseve resources.

2.3 Phase 3

The final post-processing phase involves re-assembling the POEs for the individual pathways and platforms into one matrix. The final dataframe is then saved with a timestamp. Further options involved applying a threshold and saving the updated thresholded frame directly into the fingerprint directory if required. For now we will use a threshold at 0.001.

3 Updating the pathprint R package files

Once the POE matrix has been assembled, the script `constructing POE thresholds.R` is used to determine the single chip enrichment score at which the scores of -1, 0 and 1 should be assigned. This script will also copy the threshold dataframe and the POE matrix into the fingerprint repository for the next package build. The metadata matrix should also be updated, using the script `Matrix metadata.R`. Now the package can be compiled, by running the Sweave document `compilingPackage.Rnw`, or as described in the pdf `compilingPackage.pdf`